AMENDMENTS TO THE SPECIFICATION

Amend paragraph [0017] to read as follows:

A1 Figure 14 is a flow diagram ~~illustrates~~ illustrating the processing of the process conditional component in one embodiment.

Amend paragraph [0025] to read as follows:

A2 Figure 1 is a block diagram illustrating an application program and attribute store of the execution system. Sample application 100 allows the user to view, create, and modify information relating to assets (e.g., products) that are stored in an electronic catalog. The name of the application is "asset catalog." The application comprises eight interactions: login 101, do-login 102, main-menu 103, view-asset 104, create-asset 105, do-create-asset 106, modify-asset 107, and do-modify-asset 108. When the execution system receives a request (e.g., do-create-asset), it invokes the corresponding interaction of the application to perform the behavior and ~~return~~ returns a view so that subsequent requests of the application can be made. Each interaction is defined by a series of zero or more command definitions and a view definition. Each command definition defines a command (e.g., object class) that provides a certain behavior. The do-create-asset interaction includes five command definitions: app-ctx 121, begin-tx 122, compose-asset 123, store-object 124, and end-tx 125. The do-create-asset interaction is invoked after a user specified the values of the attributes of a new asset to be added to the asset catalog. The app-ctx command retrieves the current application context of the application. The application context may be used by the interaction to access certain application-wide information, such as user profile settings. The begin-tx command indicates that a transaction for the asset catalog is beginning. The compose-asset command creates an object that identifies the value of the attributes of the asset to be added to the asset catalog. The store-object command stores an entry identified by the created object in the asset catalog. The end-tx command indicates that the transaction for the asset catalog is ending. The interaction also includes a view definition 126 that identifies that a view named "view-asset" is to be invoked to prepare a response (e.g., display page) to return to the user. The attribute store 130 contains an entry for each attribute that has been defined by an interaction of the application that has been invoked. The attribute store identifies the name of the attribute, the type of the attribute, the scope of the attribute, and the current value of the attribute. For example, the last entry in the attribute store has the name of "assetPrice," the type of "integer," the value of "500,000," and the scope of "interaction." The scope of an attribute indicates the life of the attribute. An attribute with the scope of interaction (also known as "request") has a life only within the interaction in which it is defined. An attribute with the scope of session has a life only within the current session (e.g., logon session) of the application. An attribute with the scope of application has life across executions of the application. When the execution system receives a do-create-asset request, it invokes the do-create-asset interaction. The execution system first A2 instantiates the app-ctx object defined in the interaction, prepares the object by setting its attributes based on the current values of the attribute store, performs the behavior of the object by invoking a perform method of the object, and extracts the attribute values of the object by getting the attribute values and storing them in the attribute store. The execution system then instantiates the begin-tx object defined by the interaction, prepares the object by setting its attribute values based on the current values of the attribute store, performs the behavior of the object by invoking a perform method of the object, and extracts the attribute values of the object by getting the attribute values and storing them in the attribute store. The execution system repeats this process for the compose-asset object, the store-object object, and the end-tx object. The execution system then invokes the view-asset, which in this example may be a Java server page ("JSP") that retrieves the values of the attributes of the asset from the attribute store and provides a display page showing those attribute values.

Amend paragraph [0028] to read as follows:

Figure 4 illustrates a web page for logging in that is provided by the login interaction of the maintain asset catalog application. This web page prompts the user to enter A3 their username and password and to select the submit button. When the submit button is selected, an HTTP-request message is sent that identifies the asset catalog application, the do-login interaction, and the user name and password. When the execution system receives the request, it invokes the controller to perform the do-login interaction of the asset catalog application. The do-login interaction prepares a web page to return to the client ~~systemFigure~~ system. Figure 5 illustrates a web page that specifies the menu items of the asset catalog application, which is provided by the do-login interaction. In this example, the menu items are view asset, create asset, and modify asset. When the user selects the create asset menu item, an HTTP-request message is sent that identifies the asset catalog application and the create asset interaction. When the execution system receives the request, it invokes the controller to perform the create-asset interaction of the asset catalog application. That interaction prepares a web page for entry of the attributes of an asset.

Amend paragraph [0029] to read as follows:

Figure 6 illustrates a web page for entry of asset attributes that is provided by the create-asset interaction of the asset catalog application. The web page provides various fields for the user to specify various attributes of an asset. When the user selects the create asset button, an HTTP-request message is sent that identifies the asset catalog application, the do-create-asset interaction, and the attribute values entered by the user into the web page. When the execution system receives the request, it invokes the controller to perform the do-create-asset interaction to effect ~~of~~ the creation of an asset catalog entry for the specified asset with the attribute values entered by the user.

Amend paragraph [0054] to read as follows:

Figure 18 is a flow diagram illustrating the processing of the get method component of the translator in one embodiment.~~nownow~~ This component stores the attribute value of the passed get method in the attribute store with the appropriate scope. In block 1801, the component sets the scope of the attribute associated with the selected method to the default scope (e.g., interaction). In decision block 1802, if the scope is specified in the command definition, then the component continues at block 1803, else the component continues at block 1804. In block 1803, the component sets the scope to the scope specified in the command definition. In decision block 1804, if the attribute store contains a value for the attribute of the passed get method for the specified scope, then the component continues at

A5

block 1806, else the component continues at block 1805. In block 1805, the component creates an attribute store entry for the attribute of the specified scope. In block 1806, the component invokes the get method of the object to retrieve the value of the attribute. In block 1807, the component translates the retrieved value of the attribute to the type needed by the attribute store. In block 1808, the component stores the value of the attribute in the attribute store with the specified scope and then returns.